Systems Engineering
What is Systems Engineering?

- **Systems Engineering** = definition, specification, and high-level architecture of a system which is to be realized with **multiple disciplines**, typically including electronics, mechanical, software, and possible chemical engineering.

- **Primary activities:**
  - Capturing, specifying and validating the **requirements** of the system as a whole
  - Specification of the high-level **subsystem** architecture
  - Definition of the subsystem **interfaces** and **functionality**
  - Mapping the system requirements onto the various subsystems
  - Decomposing the subsystems into the **various disciplines** – electronic, mechanical, software and chemical – and defining the abstract interfaces between those aspects
Role of the systems engineer

- In all of these activities, the systems engineers are not concerned with the design of the discipline-specific aspects of the software or the electronics, but ARE concerned with the specification of what those design aspects must achieve and how they will collaborate.
Systems Engineering

According to B.P. DOUGLASS
Systems Engineering Workflow Overview

1. System_Requirements_Capture
   - May include Requirement validation via execution and test

2. Define_High_Level_Subsystem_Architecture
   - Includes validation via execution of system-level scenarios but including subsystems

3. Map_Requirements_to_Subsystems
   - Done using <include> and <extend> relations to decompose use cases

4. Decompose_Subsystems_Into_Disciplines
   - Disciplines typically include electronic, mechanical, chemical, and software

5. Software_Specifications {Subsystem Level}

6. Electronics_Specifications {Subsystem Level}

7. Chemical_Specifications {Subsystem Level}

8. Mechanic_Specifications {Subsystem Level}

Artifacts:
- System_UseCases
- Sys_UseCase_Statechart
- Sys_UseCase_Description
- Sys_Scenarios
- Subsystem_Diagram
- Subsystem_Interfaces
- Sys_Scenarios_Elaborated
- Subsys_UseCases
- Subsys_UseCase_Statechart
- Subsys_UseCase_Description
- Subsys_Scenarios
System Requirements Capture

- Identify capabilities as Use Cases
- Use Case scenario analysis
  - Identify interactions of interest
  - Add QoS Constraints
- Use Case specification
  - Informal specification
  - Add QoS Constraints
  - Specify protocols of interaction with Statecharts or activity diagrams
  - Derive scenarios from specifications

Requirements “By example”

Requirements “By specification”
Use Cases

- A use case usually provides a textual description outlining its intent and purpose.
- Use Case Description Format:
  - Name
  - Purpose
    - May be written informally (“The purpose of the capability is to…”)
  - Description
    - May be written semi-formally (“The system shall…”)
    - May be informal
    - May be a hyperlink off to a separate document
  - Preconditions
    - What is true prior to the execution of the capability?
  - Postconditions
    - What does the system guarantee to be true after the execution of the use case?
  - Constraints
    - Additional QoS requirements or other rules for the use case
Use Case Diagram

Identifies the primary capabilities of the system.

YakityYak Deluxe ™
Cell Phone use cases.
Initialize phone scenario

Use Case: Initialize Phone
Preconditions: Telephone is off
Postconditions: Telephone is on

Description:
The «sunny day» scenario. The simplest scenario where everything works

Note: POST = Power On Self Test
Initialize phone scenario (2)

**Use Case:** Initialize Phone

**Preconditions:** Telephone is off

**Postconditions:** Telephone is on

**Description:**
POST fails. Error message is displayed on the phone display.
Make Call Use Case Statechart
Define large-scale architectural pieces of the system and their interfaces before the system is decomposed into the different design disciplines (software, electronics, mechanical, and chemical).

System architecture workflow

Define High Level Subsystem Architecture

1. Identify subsystems
2. Define subsystem responsibilities
3. Define subsystem interfaces
4. Elaborate system scenarios with subsystem architecture
5. Validate subsystem architecture by executing elaborated scenarios

Subsystem diagram

Subsystem Interface Specifications

Elaborated scenarios
A « subsystem diagram » is a class diagram whose mission is to show the subsystem architecture.
Interface Specification

Subsystem Interface Specification Description

Format:

– Operation Name and parameter list
– Parameter Description (for each parameter)
  • type
  • range
– Description
– Preconditions
  • What is true prior to the execution of the operation?
– Postconditions
  • What does the system guarantee to be true after the execution of the operation?
– Constraints
  • Additional QoS requirements or other rules for the operation (including performance requirements, if appropriate)
Transceiver interface

TransceiverSubsystem

iSend

+startSession( ):SESSION_ID
+endSession(SESSION_ID id): void
+sendMsg(MESSAGE msg): Boolean

<<interface>>

{ SESSION_ID is of type int and must be in the range of 0 (no session) to 999. }

{ startSession must execute prior to sendMsg. sendMsg requires a valid session. }

{ sendMsg shall retry every 2 seconds up to 10 times in the event of failure. }

iReceive

+getMessage( ):MESSAGE
+waitingMessages( ):int

<<interface>>

iTText

+sendText(TEXT_MESSAGE tMsg): Boolean
+getText( ):TEXT_MESSAGE

<<interface>>

{ message length limited to 1024 characters. }

<<subsystem>>

Charles André - University of Nice
iSend interface statechart

Statechart for iSend Interface. This specifies the orders and conditions for the execution of the operations.
Architecture validation

- Scenarios capture example traces of execution of the system. The UML sequence diagrams are the most popular way to capture scenarios. As we create a subsystem architecture and define the interfaces among this architectural components, we must ask ourselves if we have done it properly. The only way to truly answer that question is to validate the architecture via execution, and that is exactly what scenarios allow you to do.

- The point of a subsystem architecture, ultimately, is the efficient realization of the system requirements, so the validation of the architecture must take the requirements into account.
Use Case: Initialize Phone  
Preconditions: Telephone is off  
Postconditions: Telephone is on

Description:  
The «sunny day» scenario. The simplest scenario where everything works. This elaborated scenario shows the subsystem interaction. Original system-level terms are colored in red for clarity.
Use Case: Initialize Phone
Preconditions: Telephone is off
Postconditions: Telephone is on

Description: The «sunny day» scenario. The simplest scenario where everything works. This elaborated scenario shows the subsystem interaction. Original system-level terms are colored in red for clarity.
Use Case: Initialize Phone
Preconditions: Telephone is off
Postconditions: Telephone is on

Description:
The «sunny day» scenario. The simplest scenario where everything works. This elaborated scenario shows the subsystem interaction. Original system-level terms are colored in red for clarity.
Mapping requirements to subsystems

- **Goal**: Provide the subsystem development teams with detailed requirements specific to their subsystems, and a well-defined means for plugging their subsystem into the overall system architecture in such a way that the completed system meets all its functional and QoS requirements.

- **Primary Artifacts**: For each subsystem the following is produced: a Subsystem Use Case Model, consisting a set of use cases (and associated actors), each of which identifies or defines
  - Use Case Description
  - Use Case statechart or activity diagram (optional)
  - Use Case Scenarios
Mapping requirements

UML Diagrams and Model Elements Utilized:

– Use case Diagram, containing
  • Use Cases
  • Actor (including “internal actors,” i.e. peer subsystems)
  • Relations among model elements
    – Associations between actors and use cases
    – Generalization of actors and use cases
    – Dependency among use cases
  • Constraints
– Text Description
  • May be done in Description field associated with Use case
  • May be done in a separate documentation tool (e.g. word)
– Statechart associated with a single subsystem-level use case
– Activity diagram associated with a subsystem-level single use case
– Sequence diagram, each depicting a single scenario of a subsystem-level single use case
Hierarchical architecture model

Systems engineer’s concerns

Discipline experts
Subsystem level Use Cases

Display Subsystem
- Display Calling
- Display Power
- Display Call Status

Power Subsystem
- Power display
- Power transceiver

Phone Data Subsystem
- Lookup Number

Dialing Management Subsystem
- Initiate Session
- Maintain Session
- Dial Number

Transceiver Subsystem
- Packetize Messages
- Transmit Message
- Receive Message

Subsystem level use cases derived from System Use Cases "Make Call". All dependences here are <<includes>> (implicit on the diagram)
Subsystem Use Cases

Power Subsystem in use cases
Derived from the system level use cases and the subsystem architecture.

Power for SIM Read
Power for SIM Write
Power SIM

Power Speaker
Power Display
Monitor Power
Charge Battery

aCharger
Power Speaker
Power Transceiver
Power Display
Monitor Power
Charge Battery

aSIM
aTranceiver Subsystem
aBattery
aCharger

aSpeaker
Power Subsystem in use cases
Derived from the system level use cases and the subsystem architecture.
**Discipline decomposition**

- **Goal**: Decompose the internal structure of a subsystem into the various disciplines, define the interfaces among those design aspects, and map the requirements to those disciplines.

- **Primary Artifacts**: Subsystem Deployment Design

- **UML Diagrams and Model Elements Utilized**:
  - Deployment Diagram
  - Component Diagram
Deployment architecture

Deployment architecture for the Power Subsystem. Note that this CPU also runs other software - that is, the CPU is shared among the subsystems, but here we only show the Power Management software.